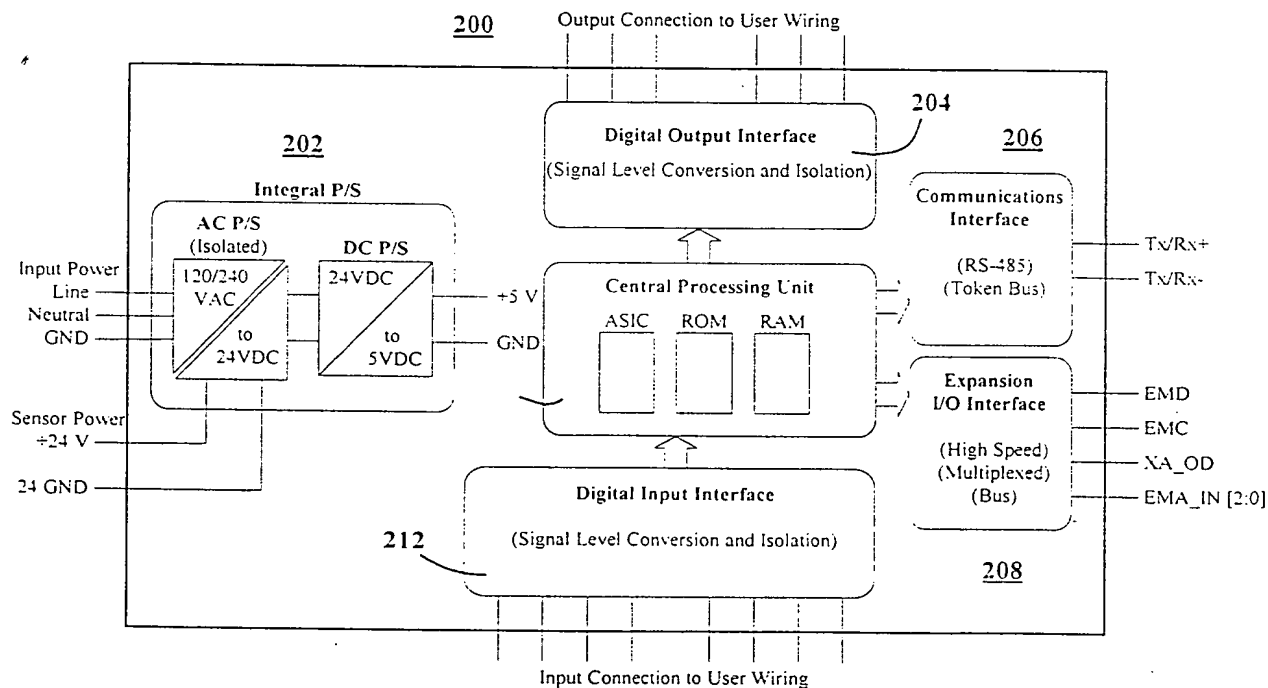
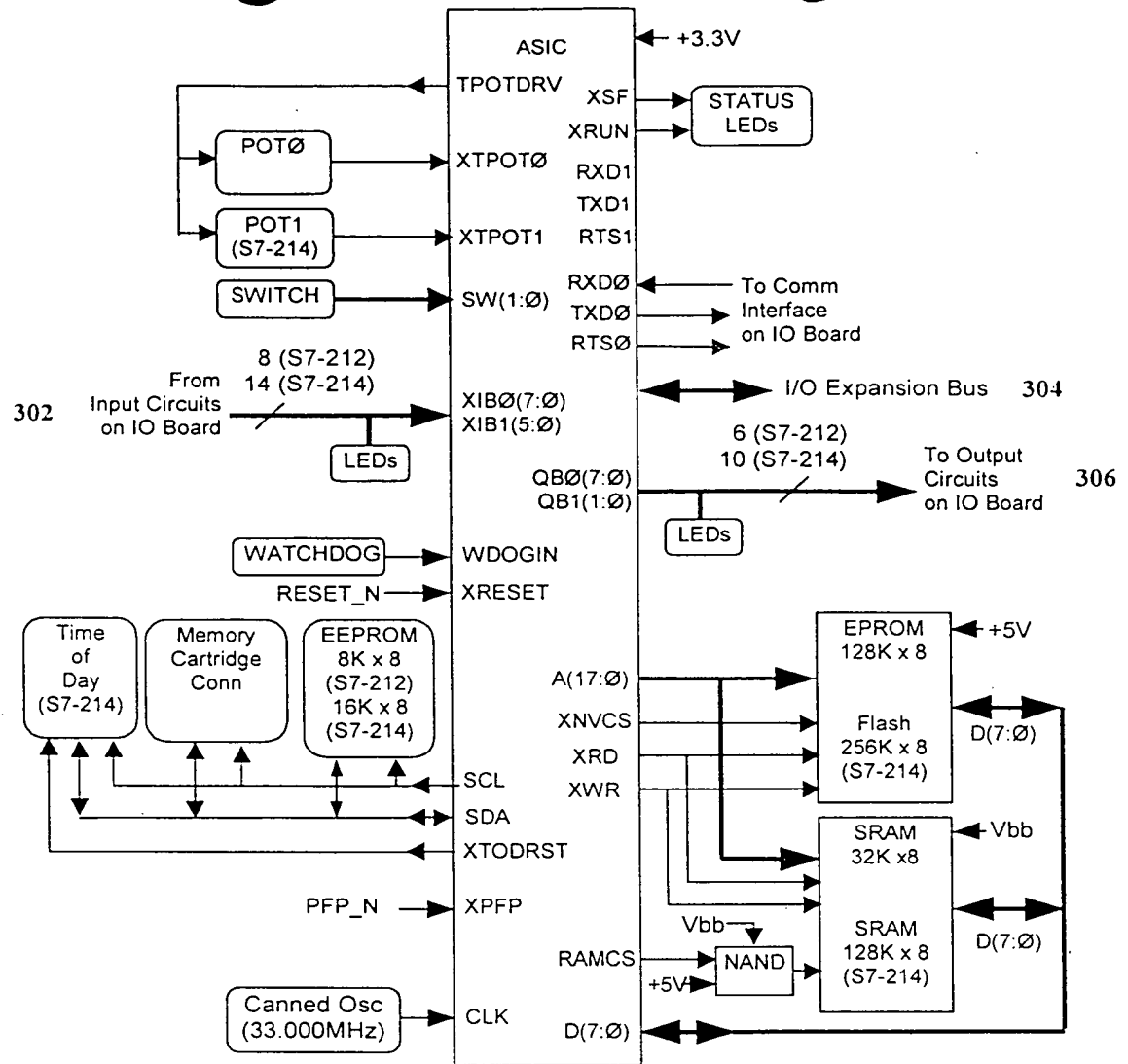


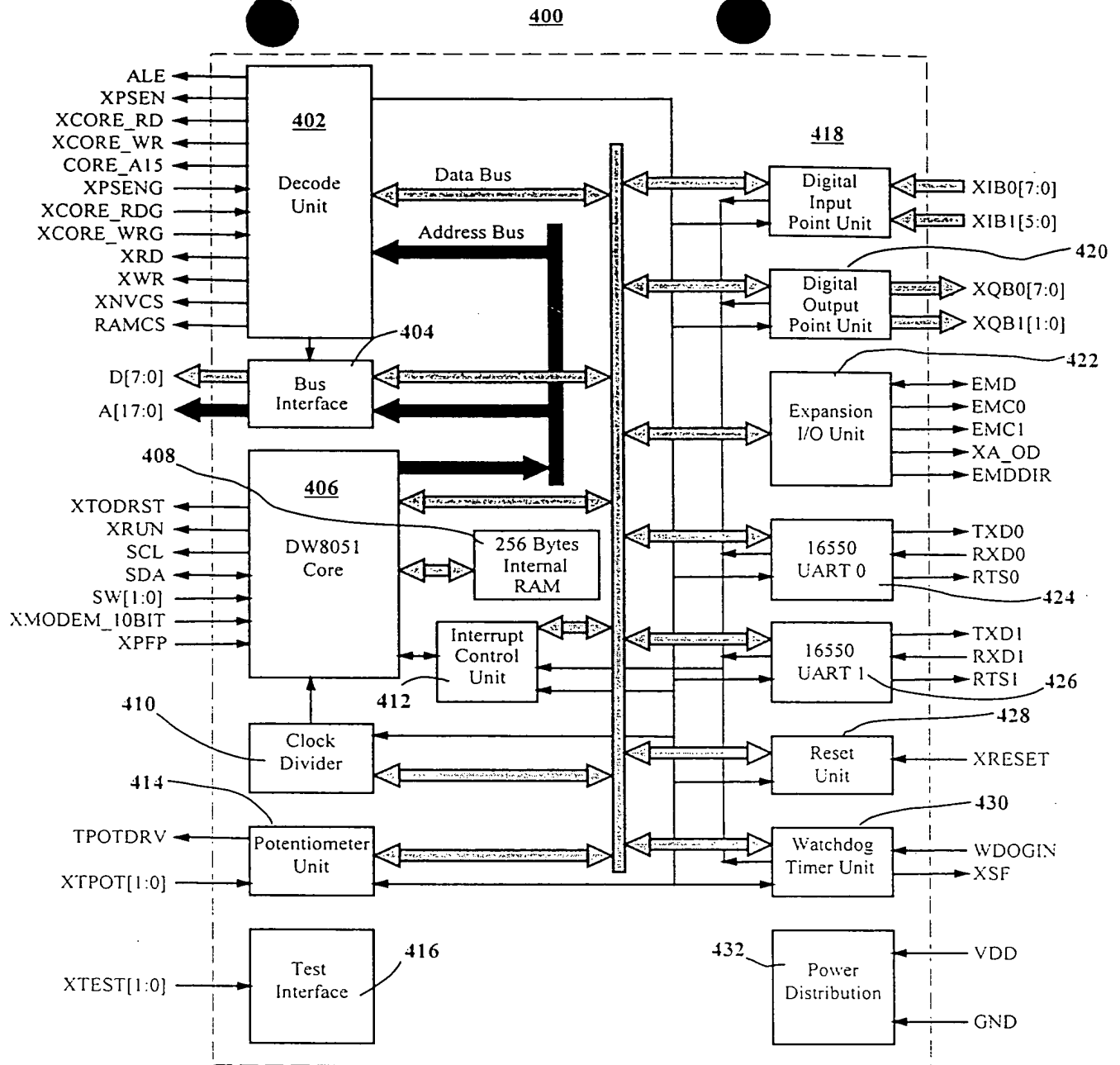
**Figure 1**



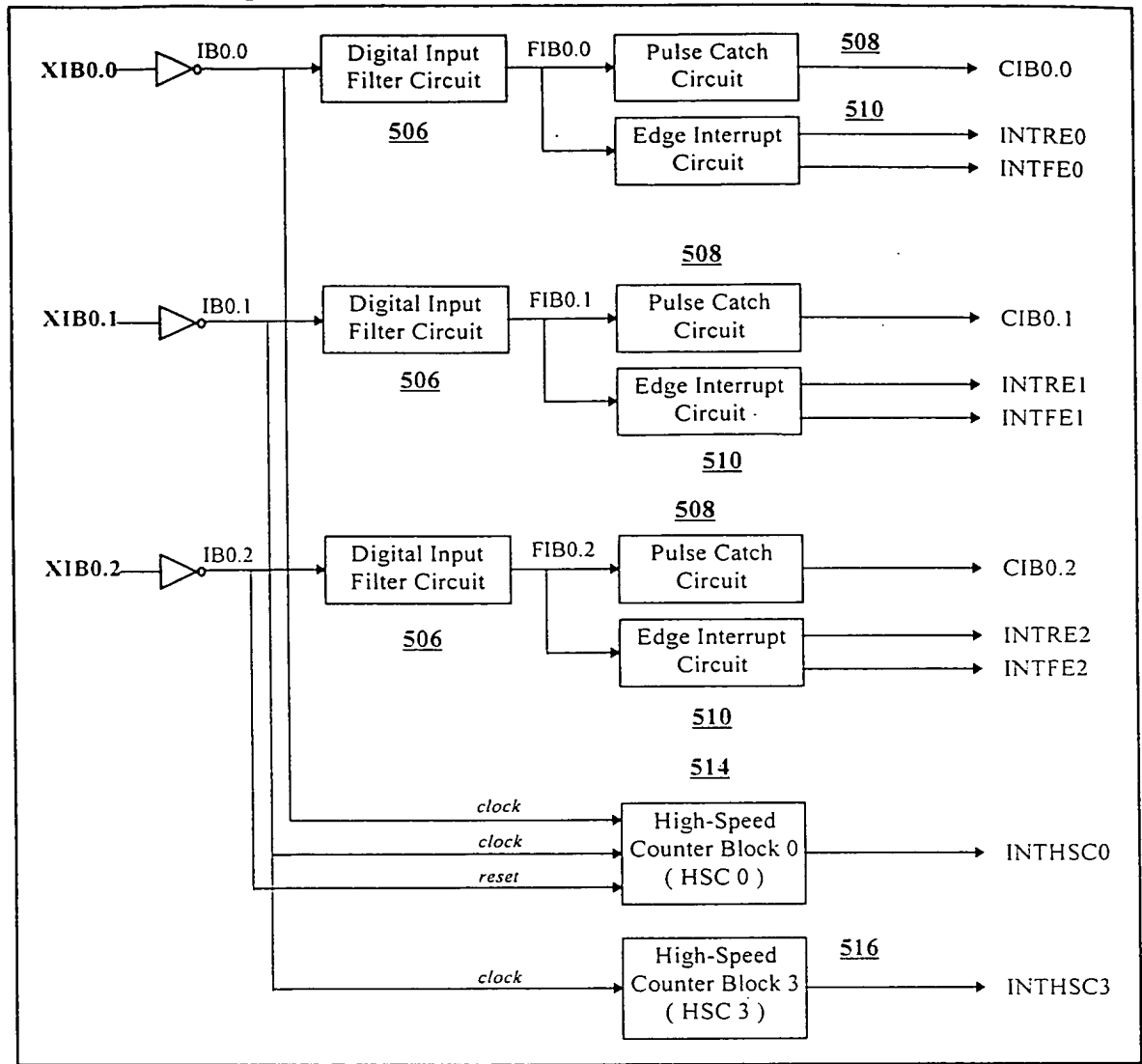
**Figure 2**



# Figure 3

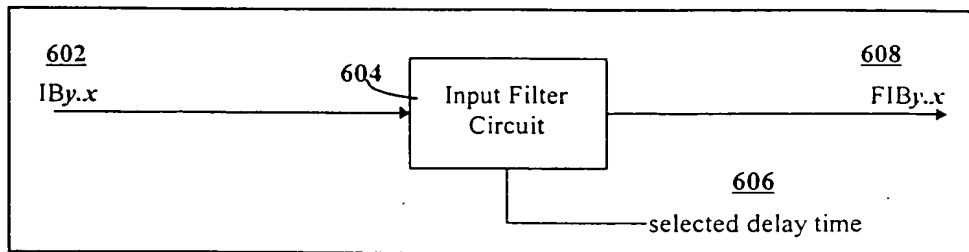


**Figure 4**



**Figure 5**

600

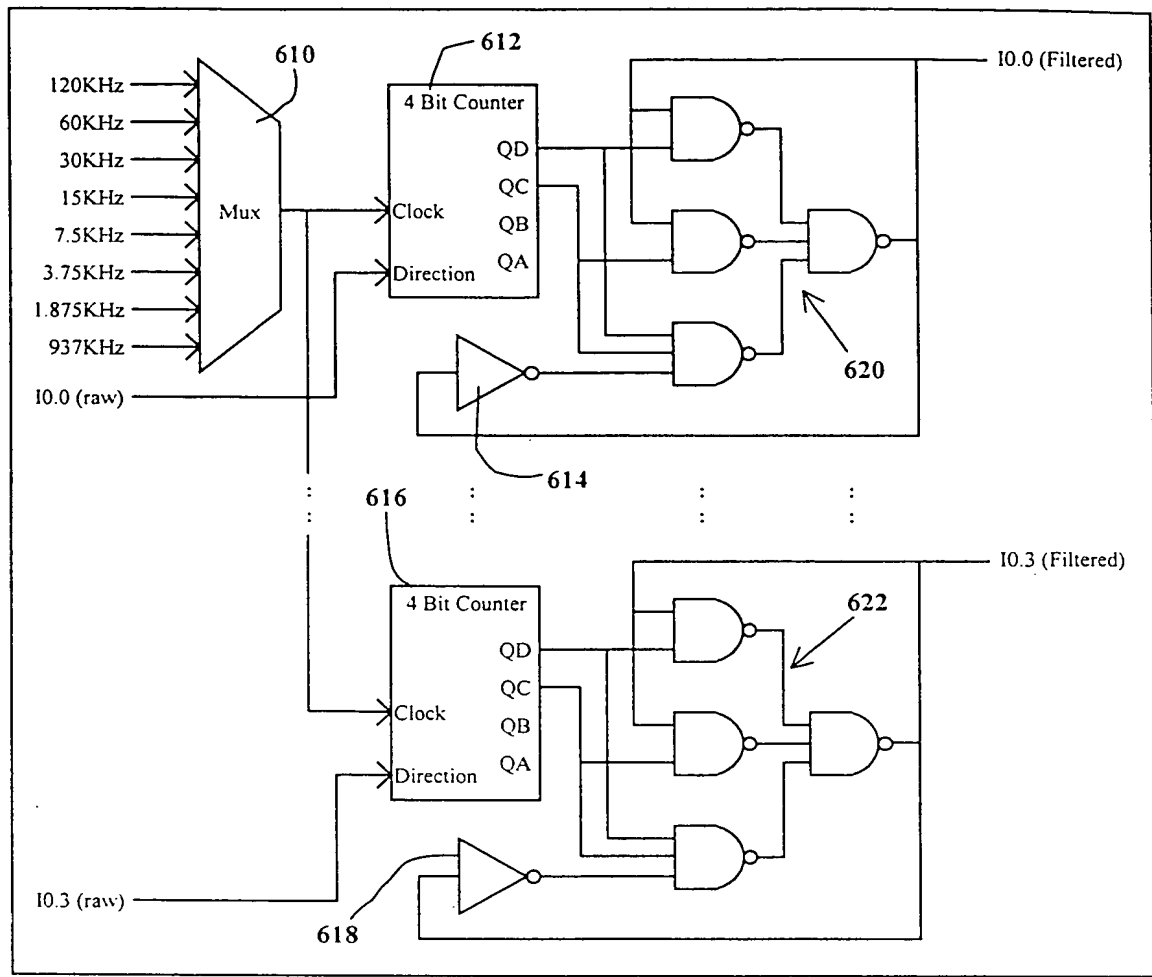


**Figure 6a**

**Digital Filter Operation Definition**

Input Point State	Current Count	Next Count	Present Output Value	Next Output Value
0 ( decrements counter )	0	0	0	0
	n, where $3 \geq n > 0$	$n - 1$	0	0
	4	3	1	0
	n, where $15 \geq n > 4$	$n - 1$	1	1
1 ( increments counter )	n, where $11 > n \geq 0$	$n + 1$	0	0
	11	12	0	1
	n, where $14 \geq n > 11$	$n + 1$	1	1
	15	15	1	1

**Figure 6b**



**Figure 6c**

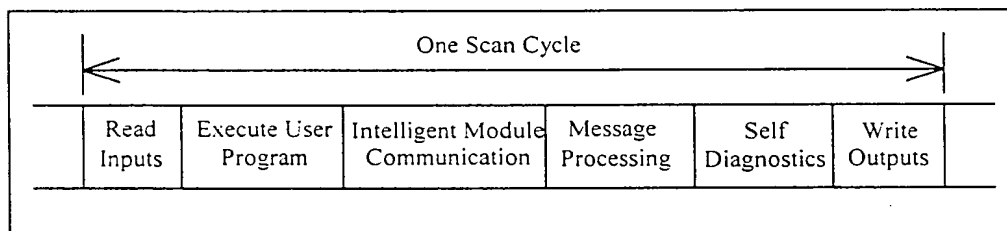
Frequency	Period	Number of Counts	Delay time
120 KHz	8.33 $\mu$ sec	12	0.1 ms
60 KHz	16.6 $\mu$ sec	12	0.2 ms
30 KHz	33.3 $\mu$ sec	12	0.4 ms
15 KHz	66.7 $\mu$ sec	12	0.8 ms
7.5 KHz	133 $\mu$ sec	12	1.6 ms
3.75 KHz	267 $\mu$ sec	12	3.2 ms
1.875 KHz	533 $\mu$ sec	12	6.4 ms
937 KHz	1067 $\mu$ sec	12	12.8 ms

**Figure 6d**

Register Value	Corresponding delay time
00	0.2 ms
01	0.4 ms
02	0.8 ms
03	1.6 ms <sup>1</sup>
04	1.6 ms <sup>1</sup>
05	3.2 ms
06	6.4 ms
07	12.8 ms
08 to FF	no delay

<sup>1</sup> Two selections for 1.6 ms delay time exist for 1st generation ASIC compatibility reasons

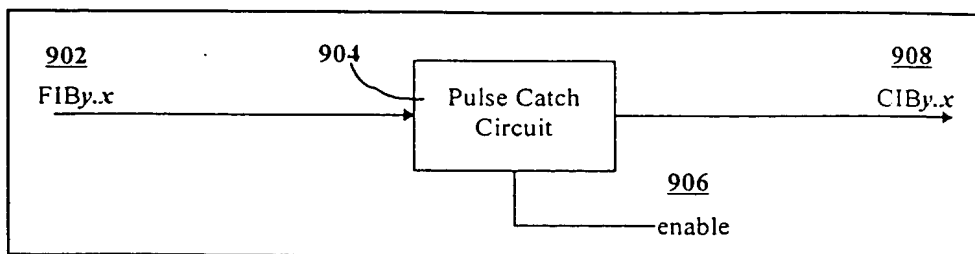
## Figure 7



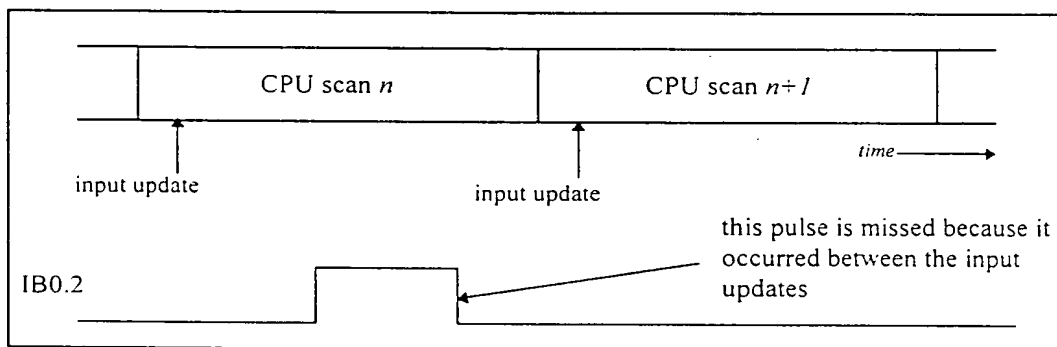
## Figure 8



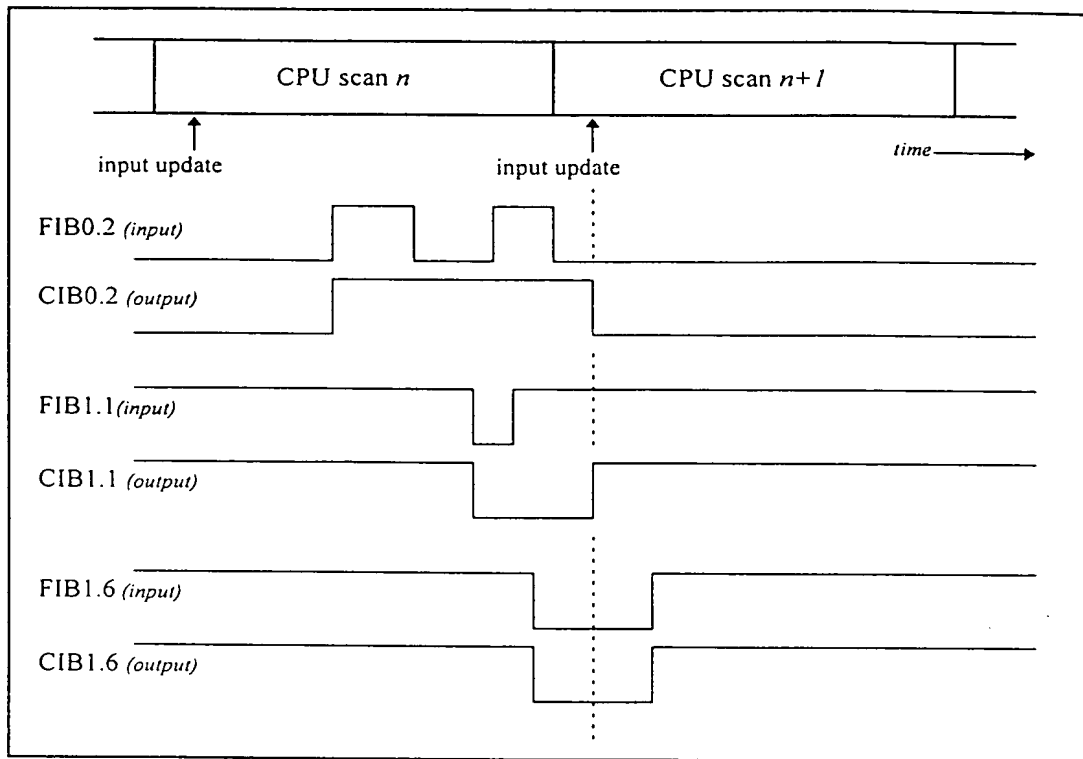
900



**Figure 9**



**Figure 10**



**Figure 11**

PCE	PS CV	PS F	RP	NS CV	NS F	Comment
1	I	0	-	CV	0	I = CV and F is not set; RP is a don't care
1	not I	0	-	I	1	I ≠ CV; capture new value of I and set F = 1
1	-	1	0	CV	1	CV has not been read
1	-	1	1	I	0	CV has been read; set CV = I
0	-	-	-	I	0	Pulse catch disabled; CV = I

**Figure 12a**

Pulse Catch Enable Registers								
Address	Description							
0002H	register name:	IB0_Pulse_Catch_Enable_Register (IB0PCE)						
	size:	byte (8-bit)						
	access:	read / write						
	reset value:	00H						
	7						0	
	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
ENx: 1 = enables pulse catch operation on input point IB0.x 0 = disables pulse catch operation on input point IB0.x								
0003H	register name:	IB1_Pulse_Catch_Enable_Register (IB1PCE)						
	size:	byte (8-bit)						
	access:	read / write						
	reset value:	00H						
	7						0	
	x	x	EN5	EN4	EN3	EN2	EN1	EN0
ENx: 1 = enables pulse catch operation on input point IB1.x 0 = disables pulse catch operation on input point IB1.x								

**Figure 12b**

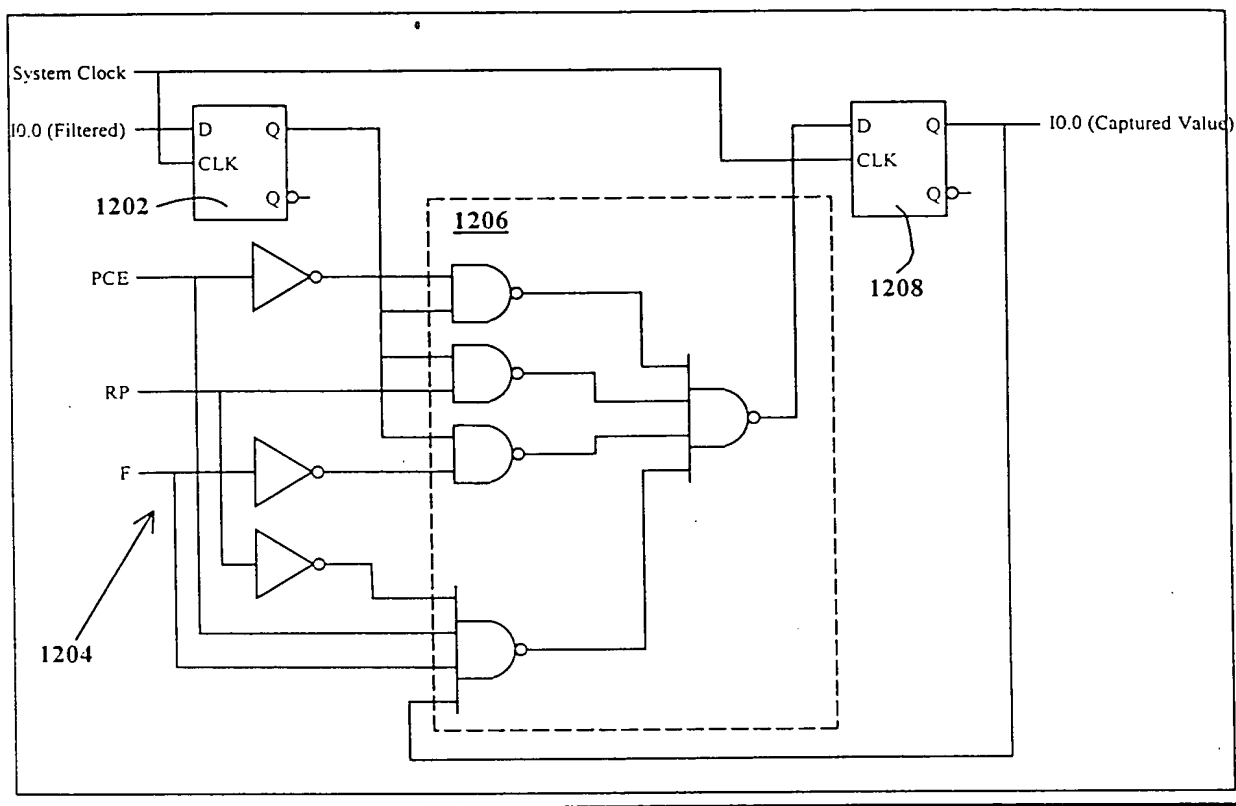
register <b>IB0PS:</b>	Read of this register returns <b>CIB0[7:0]</b> and retriggers pulse catch circuits for IB0 input points	used by SW for input update
register <b>IB1PS:</b>	Read of this register returns <b>CIB1[5:0]</b> and retriggers pulse catch circuits for IB1 input points	used by SW for input update
register <b>IB0PSNR:</b>	Read of this register returns <b>CIB0[7:0]</b> and leaves pulse catch circuits unaffected	used by SW for immediate access
register <b>IB1PSNR:</b>	Read of this register returns <b>CIB1[5:0]</b> and leaves pulse catch circuits unaffected	used by SW for immediate access

**Figure 12c**

Input Point Status Registers								
Address	Description							
0004H	register name: IB0_Input_Point_Status_Register (IB0PS)							
	size: byte (8-bit)							
	access: read only							
	reset value: 00H							
	7						0	
	CI7	CI6	CI5	CI4	CI3	CI2	CI1	CI0
CIx: Contains conditioned input point state CIB0.x								
0005H	register name: IB1_Input_Point_Status_Register (IB1PS)							
	size: byte (8-bit)							
	access: read only							
	reset value: 00H							
	7						0	
	x	x	CI5	CI4	CI3	CI2	CI1	CI0
CIx: Contains conditioned input point state CIB1.x.								
0006H	register name: IB0_Input_Point_Status_Register_No_Retrigger (IB0PSNR)							
	size: byte (8-bit)							
	access: read only							
	reset value: 00H							
	7						0	
	CI7	CI6	CI5	CI4	CI3	CI2	CI1	CI0
CIx: Contains conditioned input point state CIB0.x								
0007H	register name: IB1_Input_Point_Status_Register_No_Retrigger (IB1PSNR)							
	size: byte (8-bit)							
	access: read only							
	reset value: 00H							
	7						0	
	x	x	CI5	CI4	CI3	CI2	CI1	CI0
CIx: Contains conditioned input point state CIB1.x.								

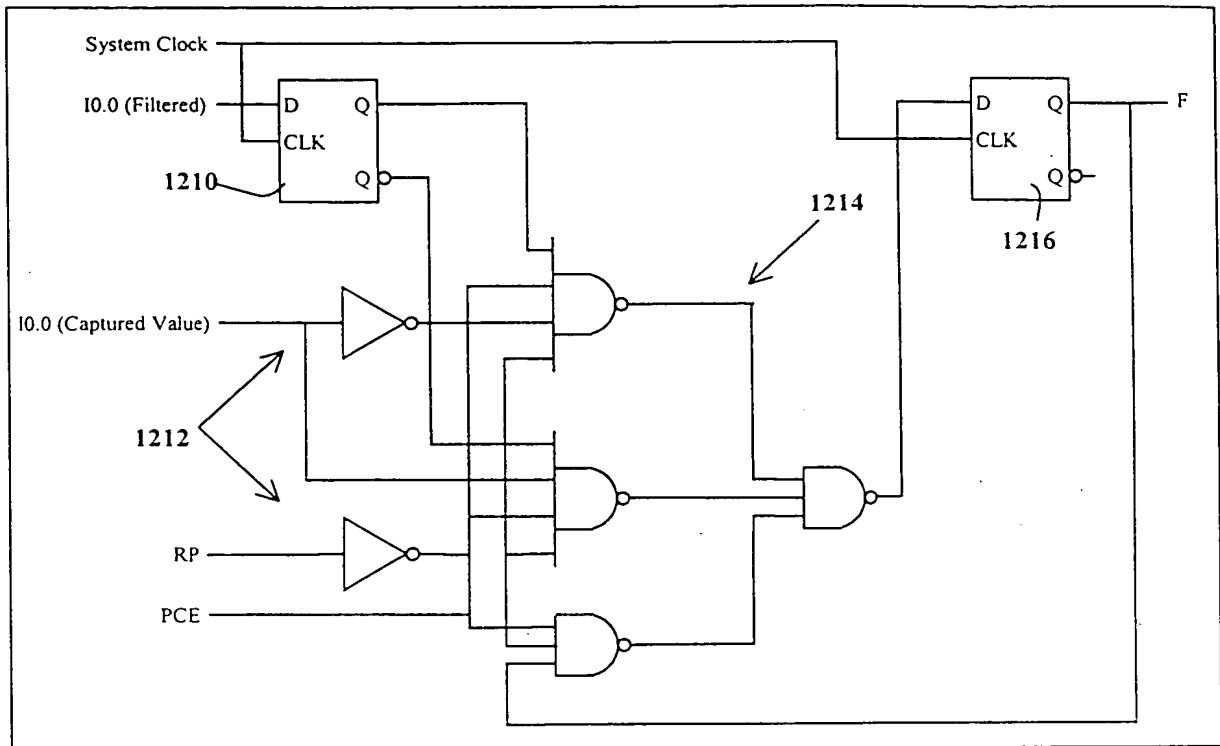
**Figure 12d**

1200e

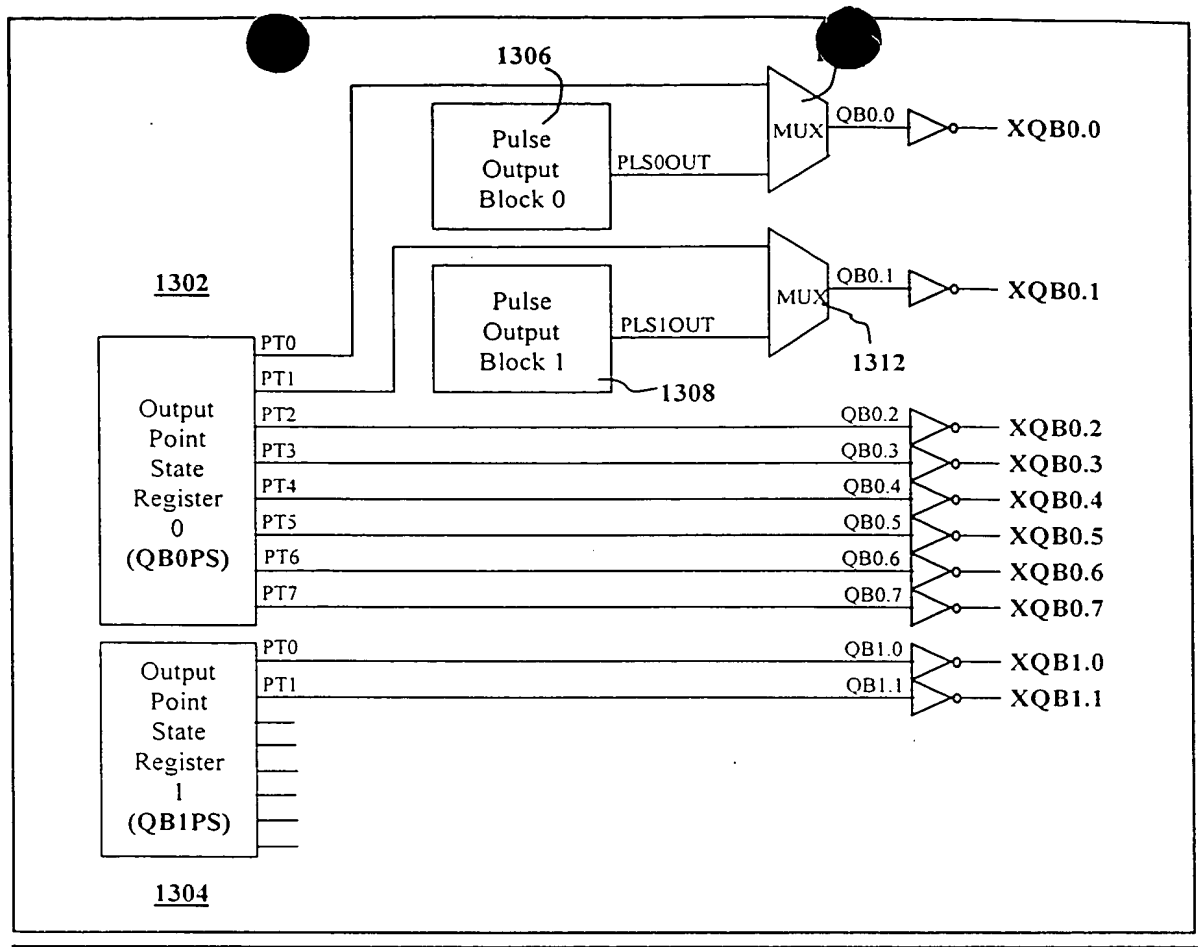


**Figure 12e**

1200f

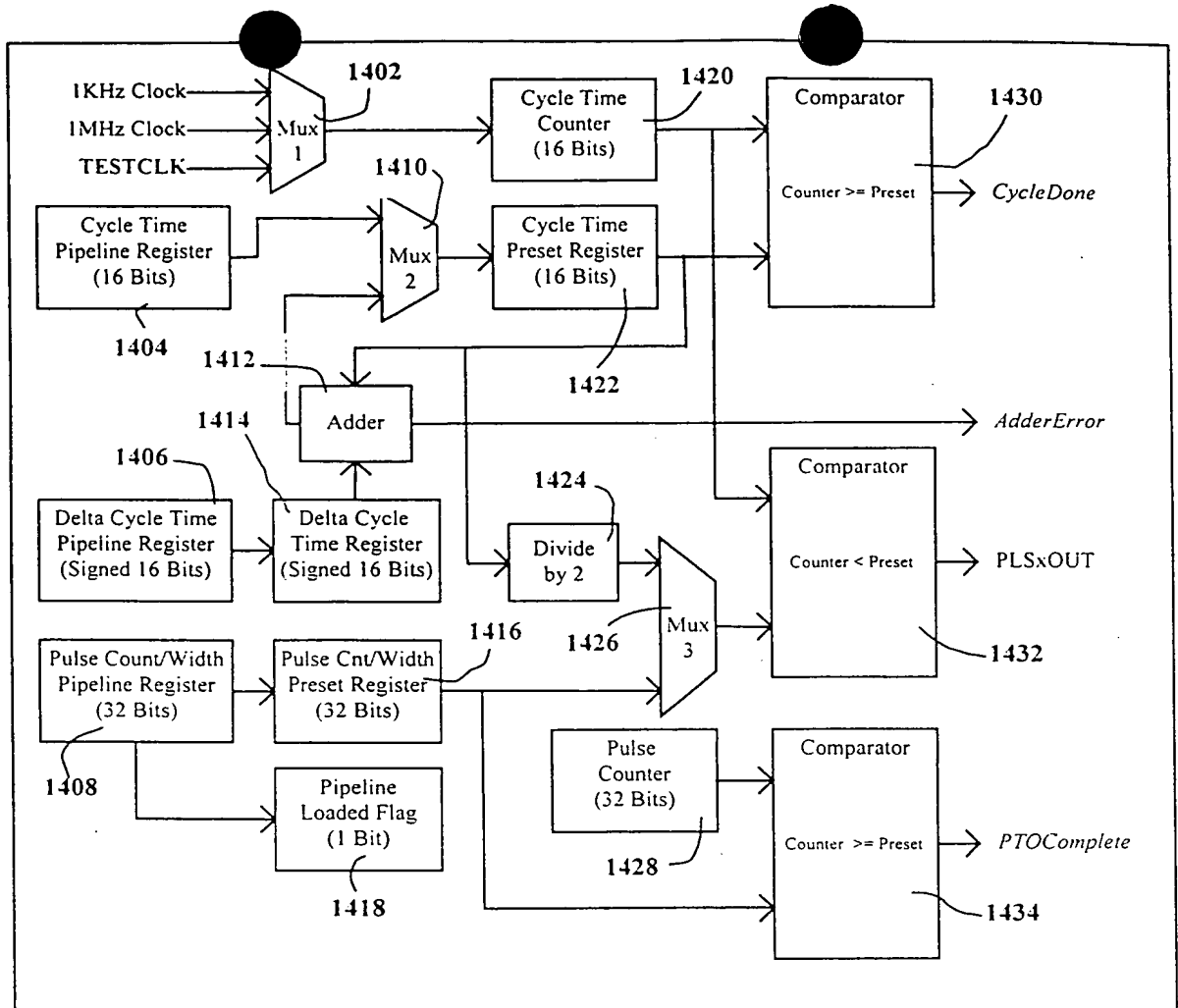


**Figure 12f**



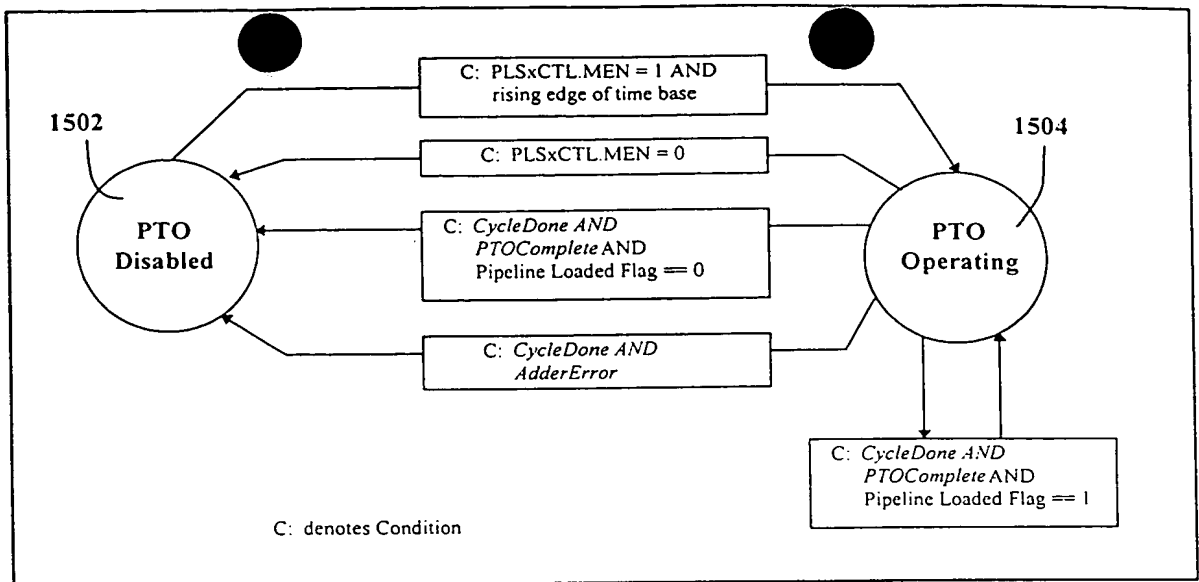
**Figure 13**





# Figure 14a





**Figure 15**

```

cycle time counter = cycle time counter + 1;
IF (cycle time counter >= cycle time preset) THEN
BEGIN // this is the CycleDone event
-----
pulse counter = pulse counter + 1;
IF (pulse counter >= pulse count preset) THEN
BEGIN // this is the PTOComplete event
-----
assert INTxPLS signal, if PTOComplete interrupts are enabled;
IF (pipeline loaded flag is set) THEN
BEGIN
-----
transfer values from pipeline registers into operating registers;
set pulse counter = 0;
clear pipeline loaded flag;
-----
END
ELSE // pipeline loaded flag is not set
BEGIN
-----
GOTO PTO Disabled state; // disable the PLS block now
-----
END
ENDIF
-----
END

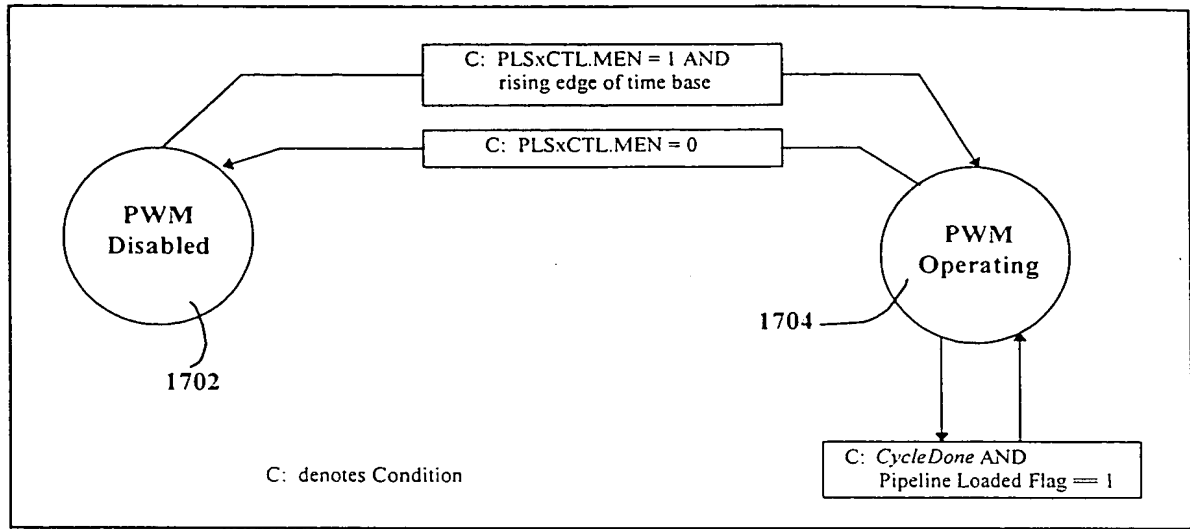
ELSE // not yet at PTOComplete
BEGIN
-----
cycle time preset = cycle time preset + delta cycle time;
IF (cycle time preset exceeds bounds) THEN
BEGIN // this is the AdderError event
-----
assert INTxPLS signal, if AdderError interrupts are enabled;
GOTO PTO Disabled state; // disable the PLS block now
-----
END
ENDIF
-----
END
ENDIF
set cycle time counter = 0;
-----
END
ENDIF

IF (cycle time counter >= (1/2 * cycle time preset)) THEN
PLSxOUT = 0;
ELSE // output still in logic high portion of the current cycle
PLSxOUT = 1;
ENDIF

```

**Figure 16**

1700



**Figure 17**

```

cycle time counter = cycle time counter + 1;
IF (cycle time counter >= cycle time preset) THEN
BEGIN // this is the CycleDone event
    -----
    IF (pipeline loaded flag is set) THEN
    BEGIN
        -----
        transfer values from pipeline registers into operating registers;
        clear pipeline loaded flag;
        -----
    END
    ENDIF
    set cycle time counter = 0;
    -----
END
ENDIF

IF (cycle time counter >= (pulse width preset)) THEN
    PLSxOUT = 0;
ELSE // output still in logic high portion of the current cycle
    PLSxOUT = 1;
ENDIF

```

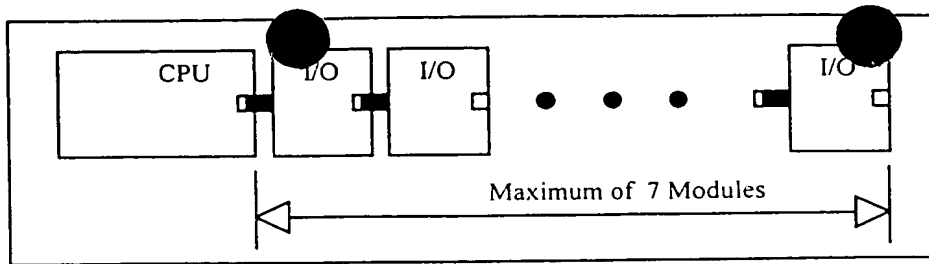
**Figure 18**

HIGH SPEED OPERATIONS				VALID OPERANDS
Instruction	Mnemonic & Operand(s)	Description	STL Status Element	
Pulse Train Output Profile	PTOP t, n	When S0 = 1, the PTO profile specified in the table for output n is executed. ENO ← S0 * /e	STK, <current step>	Enable: S0 Table: VB, IB, QB, MB, (UI) SMB, SB, LB, *VD, *AC Output: KW (UI) 0 - 1

Definition of the TABLE for PTO:

Byte Offset	Segment	Description of Table Entries
0		Number of profile segments (40 segments maximum)
1		Current step number being executed
2	#1	Number of steps (4 steps minimum)
4		Starting cycle time for this segment (2 to 65535 µsec)
6		Change in cycle time per step (signed value) (0 to 65535 µsec)
8	#2	Number of steps (4 steps minimum)
10		Starting cycle time for this segment (2 to 65535 µsec)
12		Change in cycle time per step (signed value) (0 to 65535 µsec)
:	:	:
:	:	:

**Figure 19**

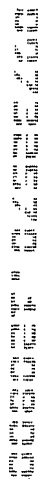


**Figure 20**

Signal Name	Use
EMD	Expansion Module Data - a bi-directional signal used to communicate the address and data to and from the module.
EMC[1:0]	Expansion Module Clocks - One clock is used to access expansion I/O external to the PLC, while the other is used to access I/O that is local to the PLC that does not connect directly to the ASIC.
XA_OD	Address/Output Disable - a dual function signal used to reset the state machine in the modules on the first clock of each access cycle (active low) and used to indicate output disable when a fatal error has been detected (active low for an RC time constant).
EMDDIR	Expansion Module Data Direction - this signal indicates the direction of data flow on the EMD signal line. 0 - Data is driven by the module to the PLC 1 - Data is driven by the PLC to the module
EMA[2:0]	Expansion Module Address - these signals are daisy chained from PLC to module to module. The value input to a module becomes that module's address. The module will output its address plus one to the next module. The PLC drives these signals to 0 so that the module connected to the PLC has the address of 0.
+5V	5 volt power supply - Two signals carry +5 volts.
GND	Power supply return - Two signals carry ground.

**Figure 21**





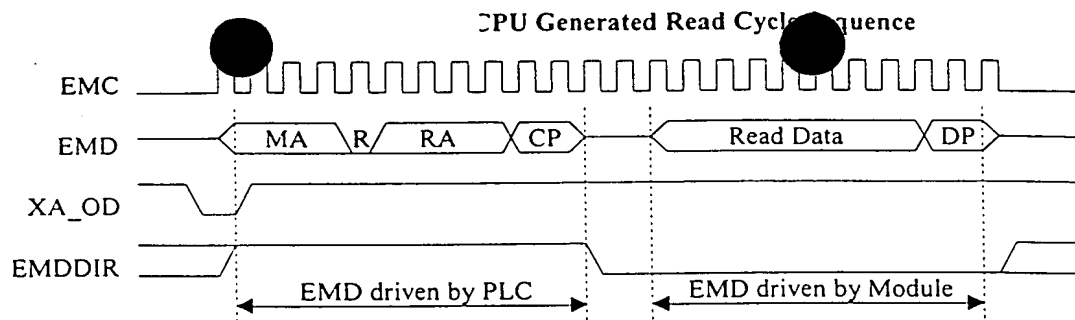
## Figure 22

Reference Designator	Component Value/Type	Component Description
IC1	74ABT125/74ABT126	Tri-state buffer
CR1	TVS 5.6V Zener	Diode
R1	4.7K ohm	Resistor
R2	110 ohm	Resistor
R3	220 ohm	Resistor
R4	22 ohm	Resistor
C1	0.1 $\mu$ F	Capacitor
C2	100pF	Capacitor

**Figure 23**

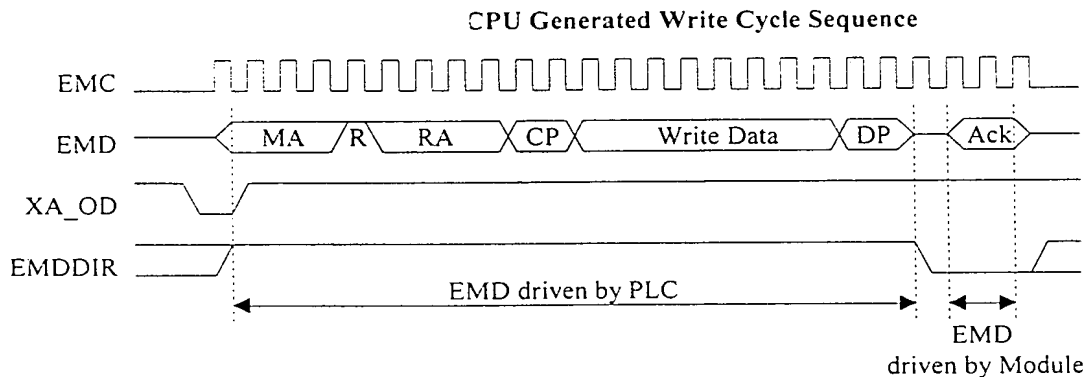
Signal Name	Driving Device	Drive Levels				Receive Levels	
		V <sub>OL</sub> (Volts) (Max)	I <sub>OL</sub> (mA) (Min)	V <sub>OH</sub> (Volts) (Max)	I <sub>OH</sub> (mA) (Min)	V <sub>IL</sub> (Volts) (Max)	V <sub>IH</sub> (Volts) (Min)
EMA[2:0]	Module	0.5	3.0	2.4	-3.0	0.8	2.0
XA_OD	CPU	0.55	64.0	2.0	-32.0	0.8	2.0
EMC	CPU	0.55	64.0	2.0	-32.0	0.8	2.0
EMD	CPU	0.55	64.0	2.0	-32.0	0.8	2.0
	Module	0.55	64.0	2.0	-32.0	0.8	2.0

**Figure 24**



MA - Module Address for modules 0 to 6 (3 bits)  
 R - Read/Write bit (Read active low)  
 RA - Register Address for registers R0 to R15 (4 bits)  
 CP - Control Parity generated by the CPU on MA, R, and RA (2 bits)  
 Read Data - Data read from the Module (8 bits)  
 DP - Data Parity generated by the module on read data (2 bits)

## Figure 25



MA - Module Address for modules 0 to 6 (3 bits)  
 R - Read/Write bit (Read active low)  
 RA - Register Address for registers R0 to R15 (4 bits)  
 CP - Control Parity generated by the CPU on MA, R, and RA (2 bits)  
 Write Data - Data written to the Module (8 bits)  
 DP - Data Parity generated by the CPU on write data (2 bits)  
 Ack - Module acknowledge of a good write cycle (2 bits)

## Figure 26

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	
	bit 0	bit 2	bit 4	bit 6	
P1	bit 1	bit 2	bit 3	bit 5	bit 7
5 bit odd parity = ! ( ( <i>a</i> xor <i>b</i> ) xor ( <i>c</i> xor <i>d</i> ) ) xor <i>e</i> )					

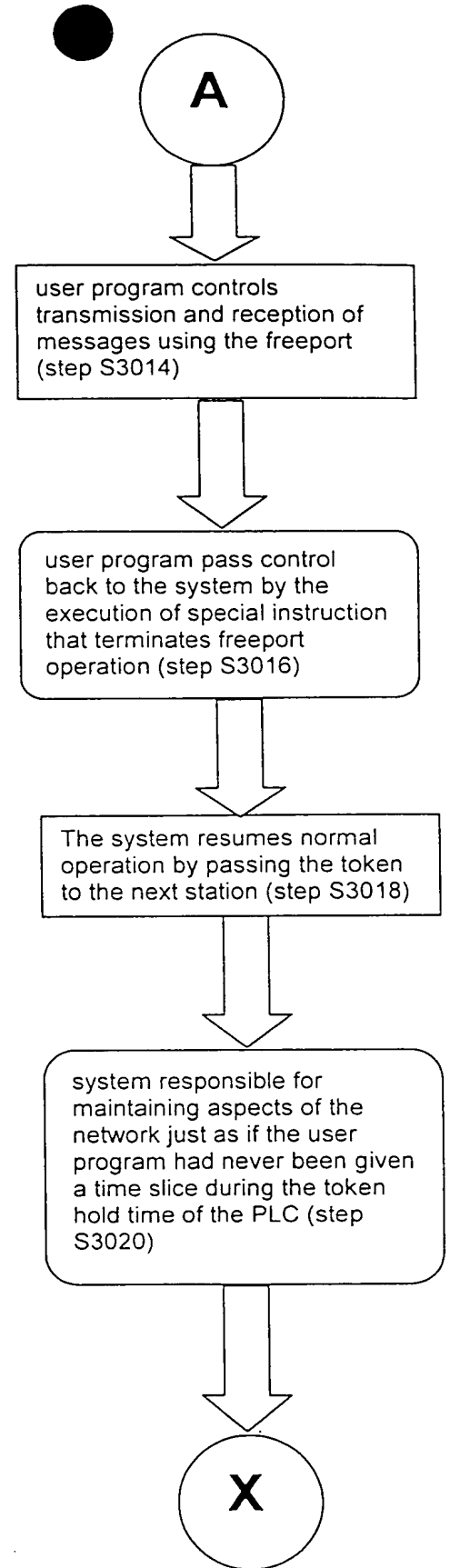
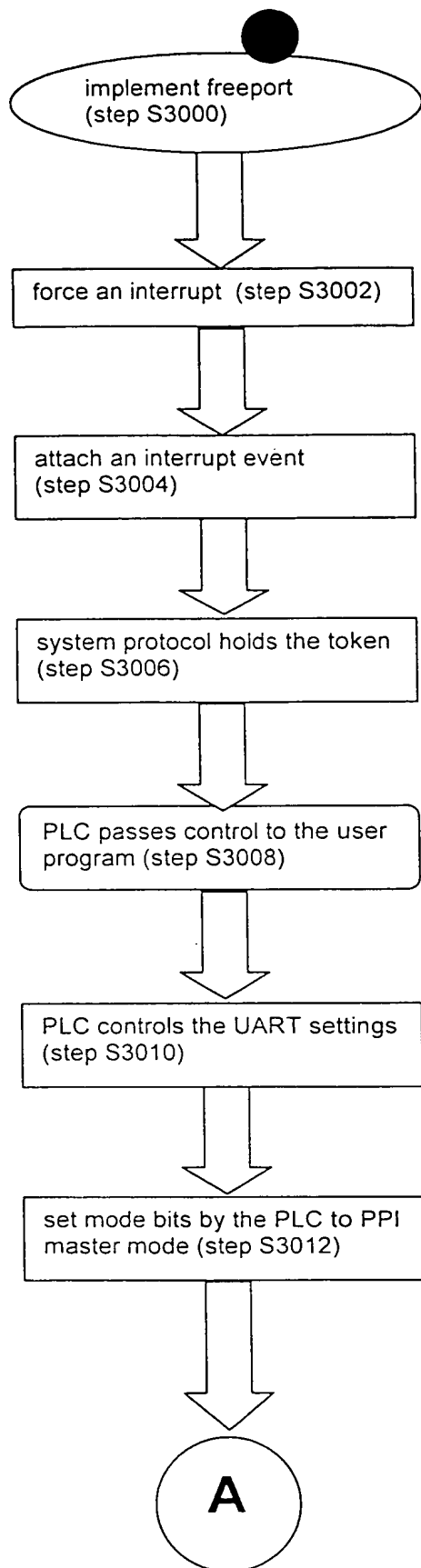
**Figure 27**

PLC Type	Port 0	Port 1
CPU 212	256	-
CPU 214	256	-
CPU 215/216	256	256

**Figure 28**

INTERRUPTS				VALID OPERANDS
Instruction	Mnemonic & Operand(s)	Description	STL Status Elements	
Pass Token	PASS	User program has completed its use of the token hold period and is returning control to the system.	STK	

**Figure 29**



**Figure 30a**

SM Bit Definition (Read/Write)			
SM Bits	Description		
SM30	<div>Port 0: Communication port usage</div> <div><div><div>MSB</div><div>7</div></div><div><div>LSB</div><div>0</div></div></div> <div>SMB30<div><div>p</div><div>p</div><div>d</div><div>r</div><div>r</div><div>r</div><div>m</div><div>m</div></div></div> <div><div><div>pp (Parity)</div><div>'00' - no parity</div><div>'01' - even parity</div><div>'10' - no parity</div><div>'11' - odd parity</div></div><div><div>d (Data bits/char)</div><div>'0' - 8 bits/char</div><div>'1' - 7 bits/char</div></div><div><div>rrr (Baud Rate)</div><div>'000' - 38,400</div><div>'001' - 19,200</div><div>'010' - 9600</div><div>'011' - 4800</div><div>'100' - 2400</div><div>'101' - 1200</div><div>'110' - 600</div><div>'111' - 300</div></div><div><div>mm (Protocol)</div><div>'00' - PPI Slave (default)</div><div>'01' - Freeport</div><div>'10' - PPI Master</div><div>'11' - reserved (PPI Slave)</div></div></div> <div><div>When the user selects the code mm = 10 (PPI Master), the PLC will become a master on the network allowing the NETR and NETW instructions to be executed. Bits 2 through 7 are ignored in PPI modes. In PPI Master mode with the token acquired interrupt enabled, these bits are used to setup the UART prior to transferring control to the user's program.</div></div>		
SM130	<div>Port 1: Communication port usage (CPU 216 only)</div> <div><div><div>MSB</div><div>7</div></div><div><div>LSB</div><div>0</div></div></div> <div>SMB130<div><div>p</div><div>p</div><div>d</div><div>r</div><div>r</div><div>r</div><div>m</div><div>m</div></div></div> <div><div><div>pp (Parity)</div><div>'00' - no parity</div><div>'01' - even parity</div><div>'10' - no parity</div><div>'11' - odd parity</div></div><div><div>d (Data bits/char)</div><div>'0' - 8 bits/char</div><div>'1' - 7 bits/char</div></div><div><div>rrr (Baud Rate)</div><div>'000' - 38,400</div><div>'001' - 19,200</div><div>'010' - 9600</div><div>'011' - 4800</div><div>'100' - 2400</div><div>'101' - 1200</div><div>'110' - 600</div><div>'111' - 300</div></div><div><div>mm (Protocol)</div><div>'00' - PPI Slave (default)</div><div>'01' - Freeport</div><div>'10' - PPI Master</div><div>'11' - reserved (PPI Slave)</div></div></div> <div><div>When the user selects the code mm = 10 (PPI Master), the PLC will become a master on the network allowing the NETR and NETW instructions to be executed. Bits 2 through 7 are ignored in PPI modes. In PPI Master mode with the token acquired interrupt enabled, these bits are used to setup the UART prior to transferring control to the user's program.</div></div>		

**Figure 30b**

SM Bit Definition (Read/Write) (continued)	
SM Bits	Description
SM87	<p>Port 0: Receive message control byte. (CPU 212/214/216)</p> <div style="text-align: center;"> <div>MSB</div> <div>7</div> <div>en</div> <div>sc</div> <div>ec</div> <div>il</div> <div>c/m</div> <div>tmr</div> <div>bk</div> <div>0</div> <div>LSB</div> <div>0</div> </div> <p>en: Enable/disable receive message bit is checked each time the RCV instruction is executed. If this bit is a "0", then the receive message function is disabled. If this bit is a "1", then the receive message function is enabled.</p> <p>sc: 0 - ignore SMB88; 1 - use the value of SMB88 to detect start of message</p> <p>ec: 0 - ignore SMB89; 1 - use the value of SMB89 to detect end of message</p> <p>il: 0 - ignore SMW90; 1 - use the value of SMW90 to detect an idle line condition<sup>1</sup></p> <p>c/m: 0 - use timer as an inter-character timer; 1 - use timer as a message timer</p> <p>tmr: 0 - ignore SMW92; 1 - terminate receive if the time period in SMW92 is exceeded</p> <p>bk: 0 - ignore break conditions; 1 - use break condition as start of message detection</p> <p><sup>1</sup>By setting the sc and bk bits to 0 and the en, il, c/m and tmr bits to 1 with an idle line timer value of zero, SMW92 will be used to time out the RCV instruction without receiving any characters. If the timer is not used (tmr = 0), then any character received will be used as start of message.</p> <p>The bits of the message interrupt control byte are used to define the criteria by which the message is identified. Both start of message and end of message criteria are defined. To determine the start of a message either of two sets of logically Anded start of message criteria must be true and must occur in sequence (idle line followed by start character or break followed by start character). To determine the end of a message the enabled end of message criteria are logically ORed. The equations for start and stop criteria are given below:</p> <p style="text-align: center;">Start of Message = il * sc + bk * sc</p> <p style="text-align: center;">End of Message = ec + tmr + maximum character count reached</p> <p>Note: Receive will automatically be terminated by an overrun or a parity error (if enabled).</p>
SM88	Port 0: Start of message character. (CPU 212/214/216)
SM89	Port 0: End of message character. (CPU 212/214/216)
SM90	Port 0: Idle line time period given in milliseconds. The first character received after the idle line time has expired is the start of a new message. SM90 is MSB.
SM91	(CPU 212/214/216)
SM92	Port 0: Inter-character/message timer timeout value given in milliseconds. If the time period is exceeded, the receive message is terminated. SM92 is MSB. (CPU 212/214/216)
SM93	
SM94	Port 0: Maximum number of characters to be received (1 to 255 bytes) (CPU 212/214/216)

**Figure 30c**

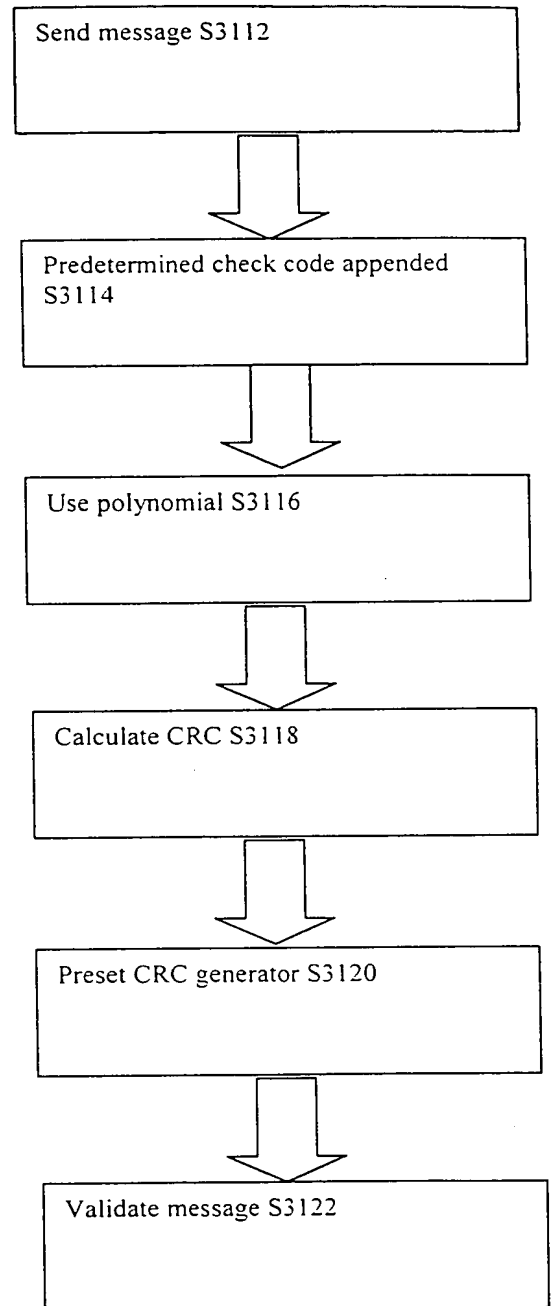
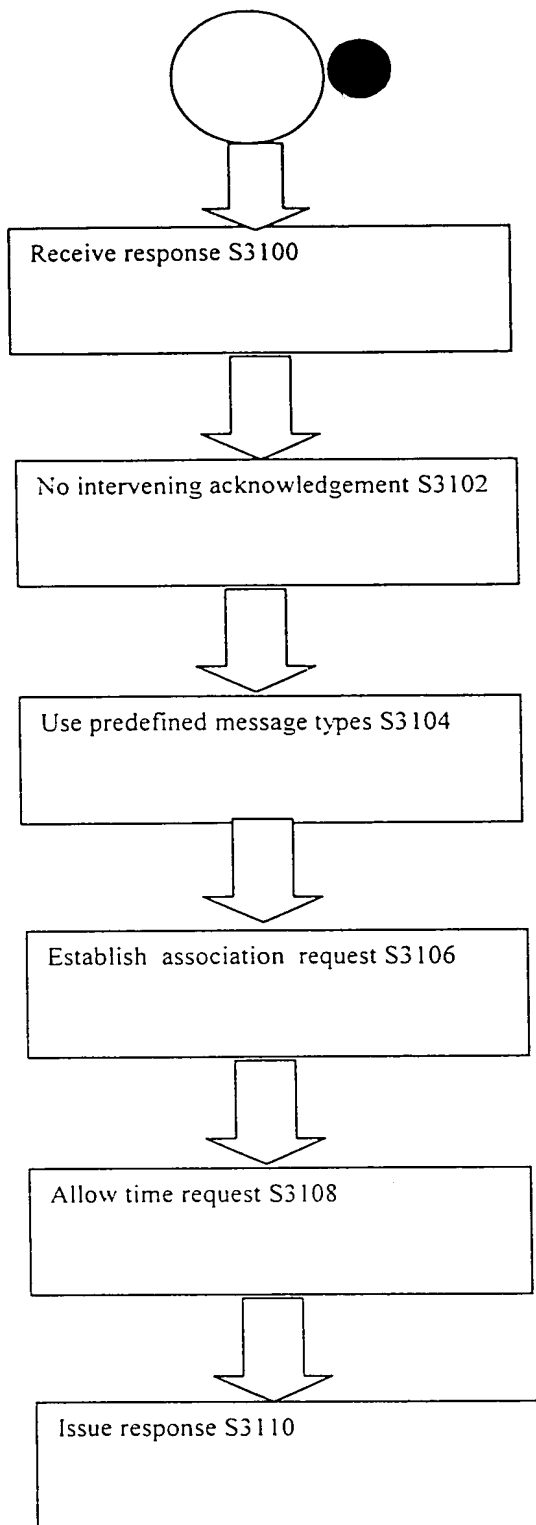
SM Bit Definition (Read/Write) (continued)	
SM Bits	Description
SM187	<p>Port 1: Message interrupt control byte (CPU 216 only)</p> <div style="text-align: center;"> <div>MSB</div> <div>7</div> <div>en</div> <div>sc</div> <div>ec</div> <div>il</div> <div>c/m</div> <div>tmr</div> <div>bk</div> <div>0</div> <div>LSB</div> <div>0</div> </div> <p>en: Enable/disable receive message bit is checked each time the RCV instruction is executed. If this bit is a "0", then the receive message function is disabled. If this bit is a "1", then the receive message function is enabled.</p> <p>sc: 0 - ignore SMB188; 1 - use the value of SMB188 to detect start of message</p> <p>ec: 0 - ignore SMB89; 1 - use the value of SMB189 to detect end of message</p> <p>il: 0 - ignore SMW190; 1 - use the value of SMW190 to detect an idle line condition<sup>1</sup></p> <p>c/m: 0 - use timer as an inter-character timer; 1 - use timer as a message timer</p> <p>tmr: 0 - ignore SMW192; 1 - terminate receive if the time period in SMW192 is exceeded</p> <p>bk: 0 - ignore break conditions; 1 - use break condition as start of message detection</p> <p><sup>1</sup>By setting the sc and bk bits to 0 and the en, il, c/m and tmr bits to 1 with an idle line timer value of zero, SMW192 will be used to time out the RCV instruction without receiving any characters. If the timer is not used (tmr = 0), then any character received will be used as start of message.</p> <p>The bits of the message interrupt control byte are used to define the criteria by which the message is identified. Both start of message and end of message criteria are defined. To determine the start of a message either of two sets of logically Anded start of message criteria must be true and must occur in sequence (idle line followed by start character or break followed by start character). To determine the end of a message the enabled end of message criteria are logically Ored. The equations for start and stop criteria are given below:</p> $\text{Start of Message} = \text{il} * \text{sc} + \text{bk} * \text{sc}$ $\text{End of Message} = \text{ec} + \text{tmr} + \text{maximum character count reached}$ <p>Note: Receive will automatically be terminated by an overrun or a parity error (if enabled).</p>
SM188	Port 1: Start of message character (CPU 216 only)
SM189	Port 1: End of message character (CPU 216 only)
SM190	Port 1: Idle line time period given in milliseconds. The first character received after the idle line time has expired is the start of a new message. SM190 is MSB. (CPU 216 only)
SM191	
SM192	Port 1: Inter-character/message timer timeout value given in milliseconds. If the time period is exceeded, the receive message is terminated. SM192 is MSB. (CPU 216 only)
SM193	
SM194	Port 1: Maximum number of characters to be received (1 to 255 bytes) (CPU 216 only)

**Figure 30d**



PLC Type	Port 0			Port 1		
	Number of Connections		Buffer Size	Number of Connections		Buffer Size
	Total	Reserved	(Bytes)	Total	Reserved	(Bytes)
CPU 221	4	1 - PG 1 - OP	128/256	-	-	-
CPU 222	4	1 - PG 1 - OP	128/256	-	-	-
CPU 224	4	1 - PG 1 - OP	128/256	-	-	-
CPU 226	4	1 - PG 1 - OP	128/256	4	1 - PG 1 - OP	128/256

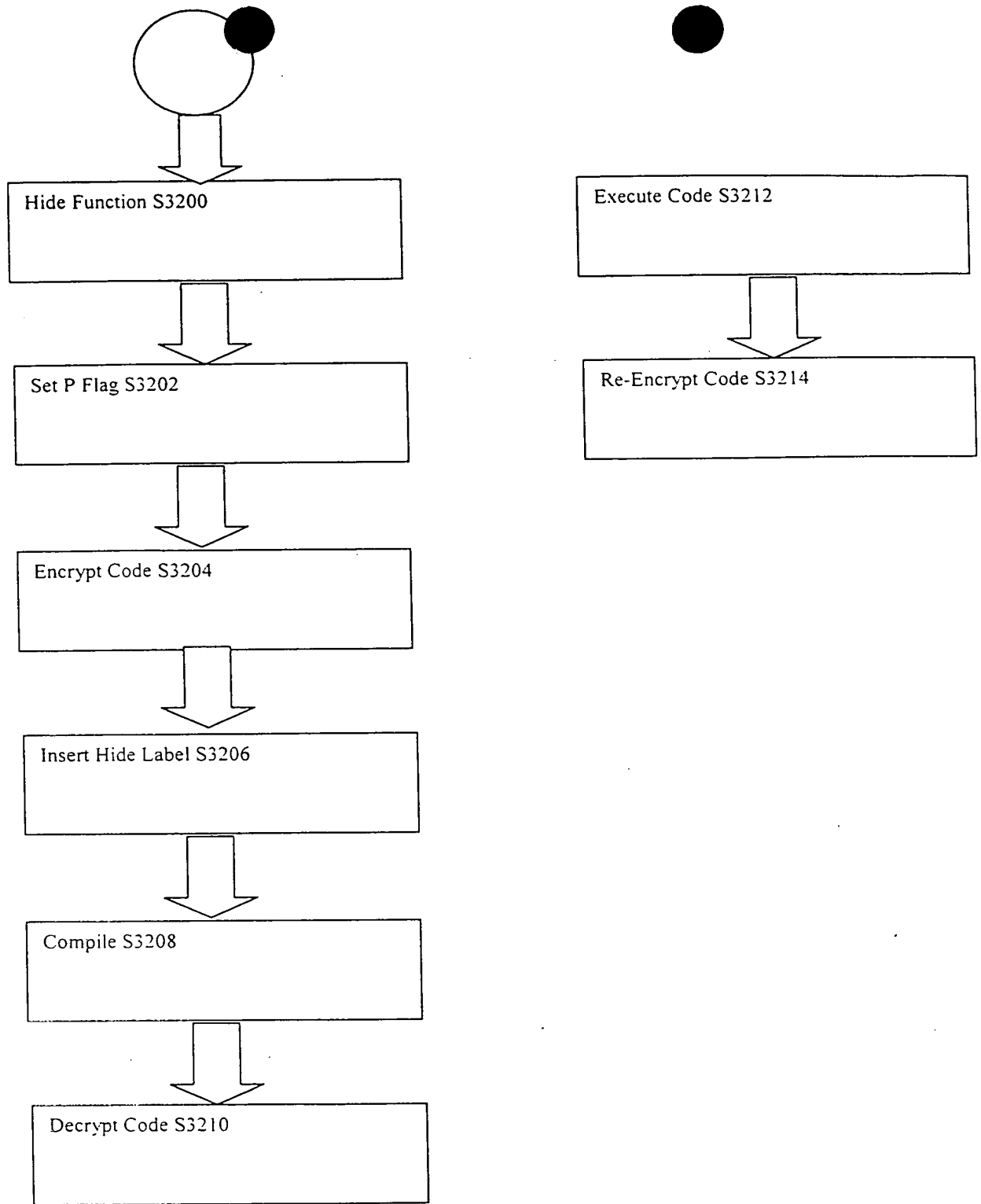
**Figure 30e**



**Figure 31**

PROGRAM CONTROL FUNCTIONS				
Instruction	Mnemonic & Operand(s)	Description	STL Status Element	VALID OPERANDS
Hide	HIDE n, a, p	The HIDE label marks the start of a block of n instructions that are encrypted using the password, p, when a is non-zero.	STK, SMB1	Enable: None n: KW (2 bytes) (UI) a: KW (2 bytes) (UI) p: KD (4 bytes) (UI)

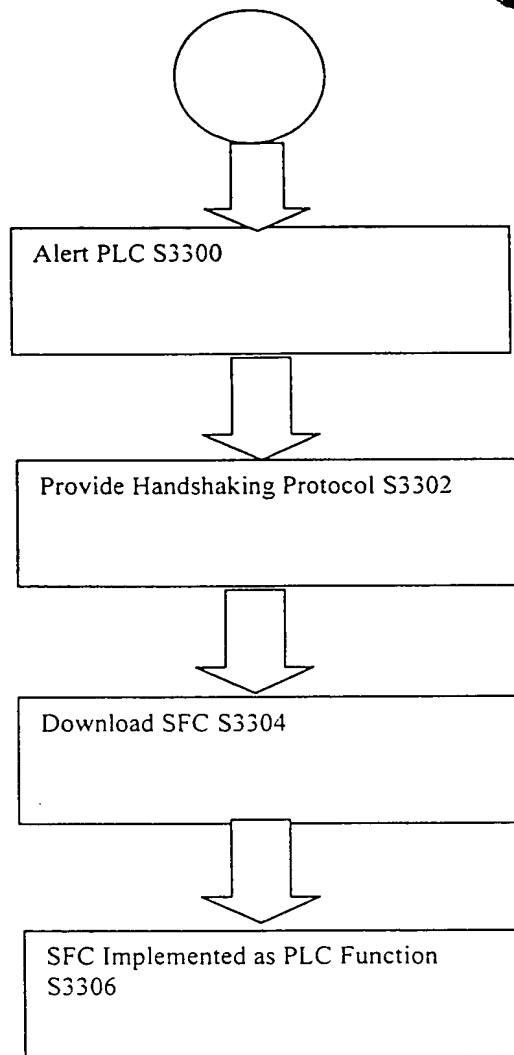
**Figure 32a**



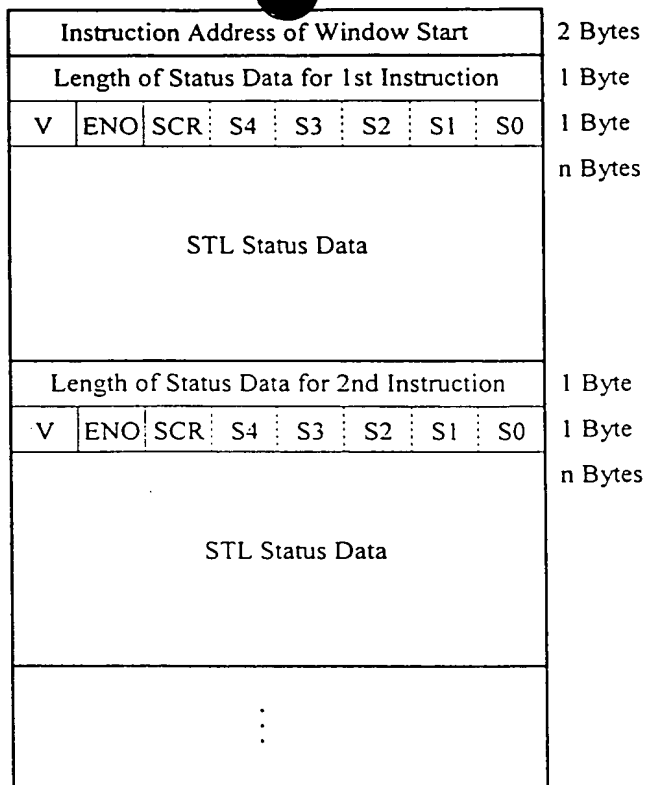
**Figure 32b**

PROGRAM CONTROL FUNCTIONS				
Instruction	Mnemonic & Operand(s)	Description	STL Status Element	VALID OPERANDS
System Function Call	SFC f, s, #_parms, p0, p1, p2, ...	When S0 = 1, execute the system function identified by f and the sub- function identified by s.	STK, SMB I, <p0>, <p1>, <p2>, ...	<p>Enable: S0 f: KW (UI) 0-65536 s: KW (UI) 0-65536</p> <p>#parms: KB (UI) 0-16</p> <p>p0, ... : Bit, Byte, Word, Dword</p> <p>Bit V, I, Q, M, SM, S, T, C, L</p> <p>Byte VB, IB, QB, MB, SMB, SB, KB, LB</p> <p>Word VW, T, C, IW, QW, MW, SMW, SW, LW, KW</p> <p>Dword VD, ID, QD, MD, SMD, SD, HC, LD, KD, &amp;VB, &amp;IB, &amp;QB, &amp;MB, &amp;T, &amp;C, &amp;SB</p> <p>Note: Constants and address pointer specifications are not allowed for output or input/output parameters.</p>

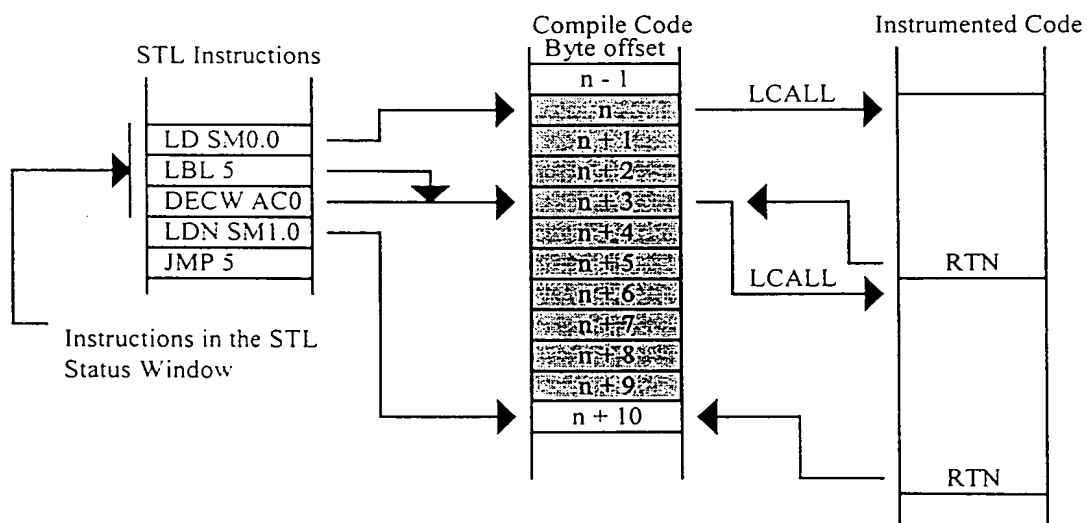
**Figure 33a**



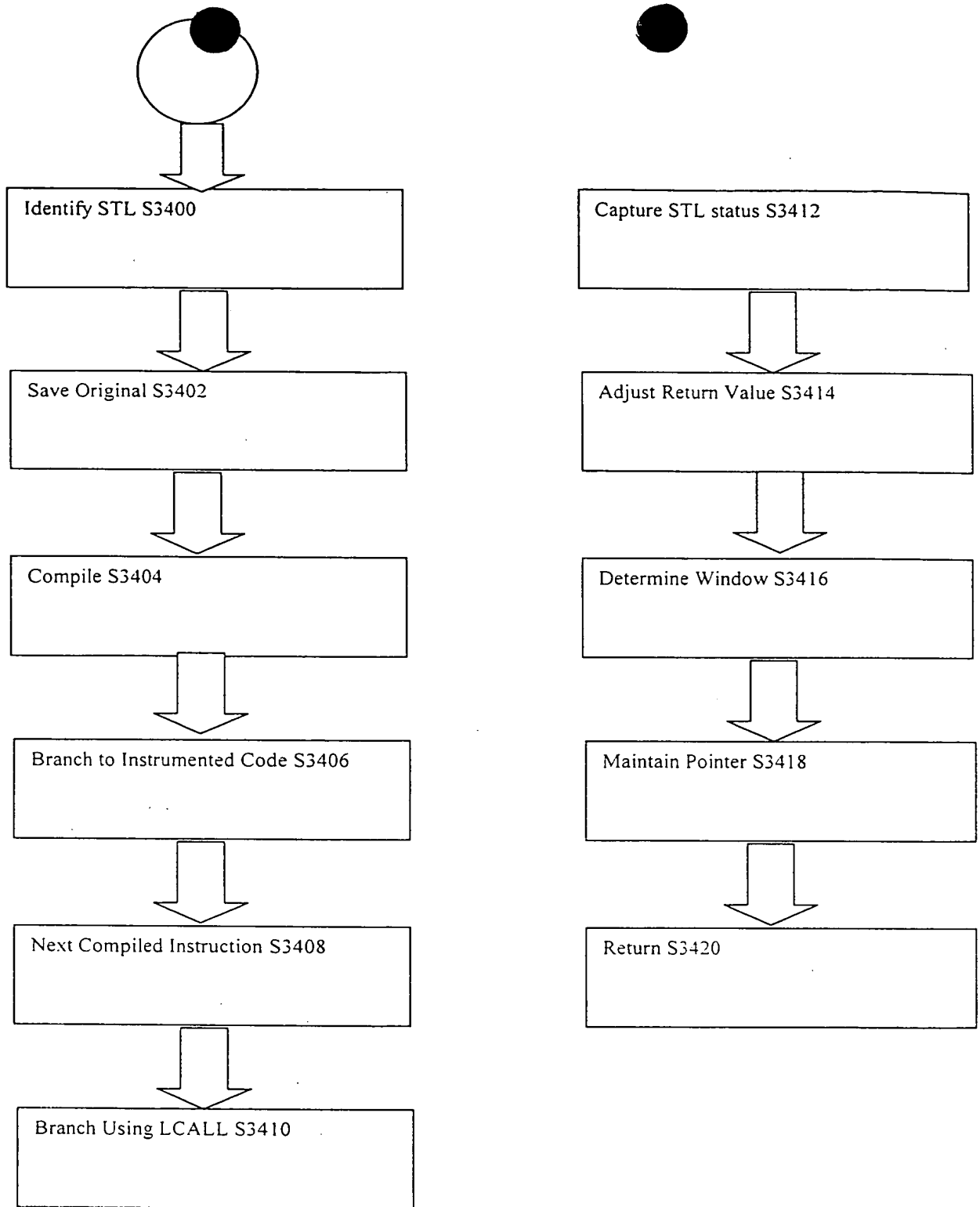
**Figure 33b**



**Figure 34a**



**Figure 34b**



**Figure 34c**

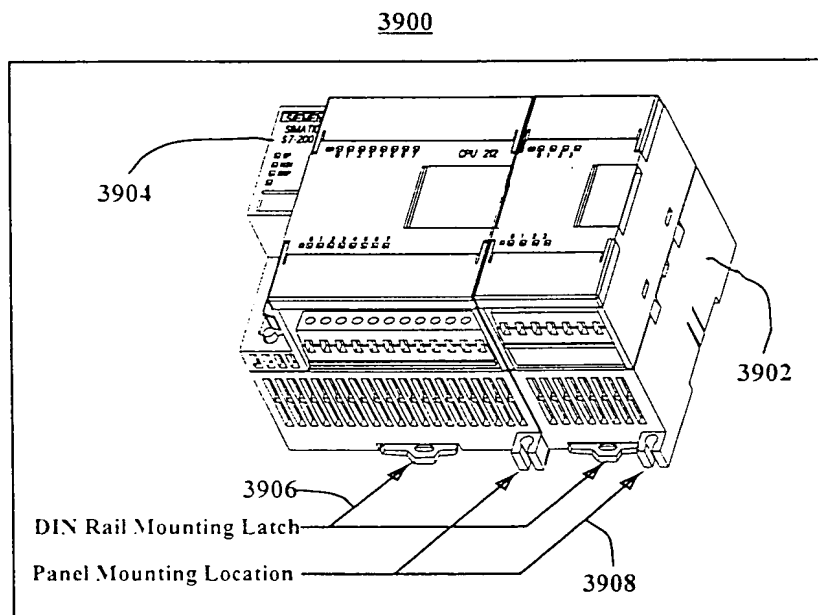


	Basic Instr.	Control To	Bytes	Cycles
1	LD	RLC A MOV C, <bit>	1 2	1 1
2	LDN	RLC A MOV C, <bit> CPL C	1 2 1	1 1 1
3	AND	ANL C, <bit> NOP	2 1	2 1
4	ANA	ANL C, <bit> NOP	2 1	2 1
5	OR	ORL C, <bit> NOP	2 1	2 1
6	ORA	ORL C, <bit> NOP	2 1	2 1
7		MOV <bit>, C NOP	2 1	2 1
8	ALD	ANL C, ACC.0 RR A	2 1	2 1
9	OLD	ORL C, ACC.0 RR A	2 1	2 1
10	NOT	CPL C NOP NOP	1 1 1	1 1 1
11	LPS	RL A MOV ACC.0, A	1 2	1 2
12	LPP	RRC A NOP NOP	1 1 1	1 1 1
13	LRD	MOV C, ACC.0	2	2
14	RET	RET NOP NOP	1 1 1	2 1 1
15	INT	CLR A NOP NOP	1 1 1	1 1 1
16	END	JNC \$+0 RET	2 1	2 2
17	CRETI	JNC \$+0 RET	2 1	2 2

**Figure 34d**

Housing	Height	Width	Depth
CPU 221	80 mm	90 mm	62 mm
CPU 222	80 mm	90 mm	62 mm
CPU 224	80 mm	120.5 mm	62 mm
CPU 226	80 mm	190 mm	62 mm
8 Point I/O Module	80 mm	46 mm	62 mm
16 Point I/O Module	80 mm	71.2 mm	62 mm
32 Point I/O Module	80 mm	125 mm	62 mm
Intelligent Module	80 mm	90 mm	62 mm

**Figure 35a**



**Figure 35b**

3610

3615

3620

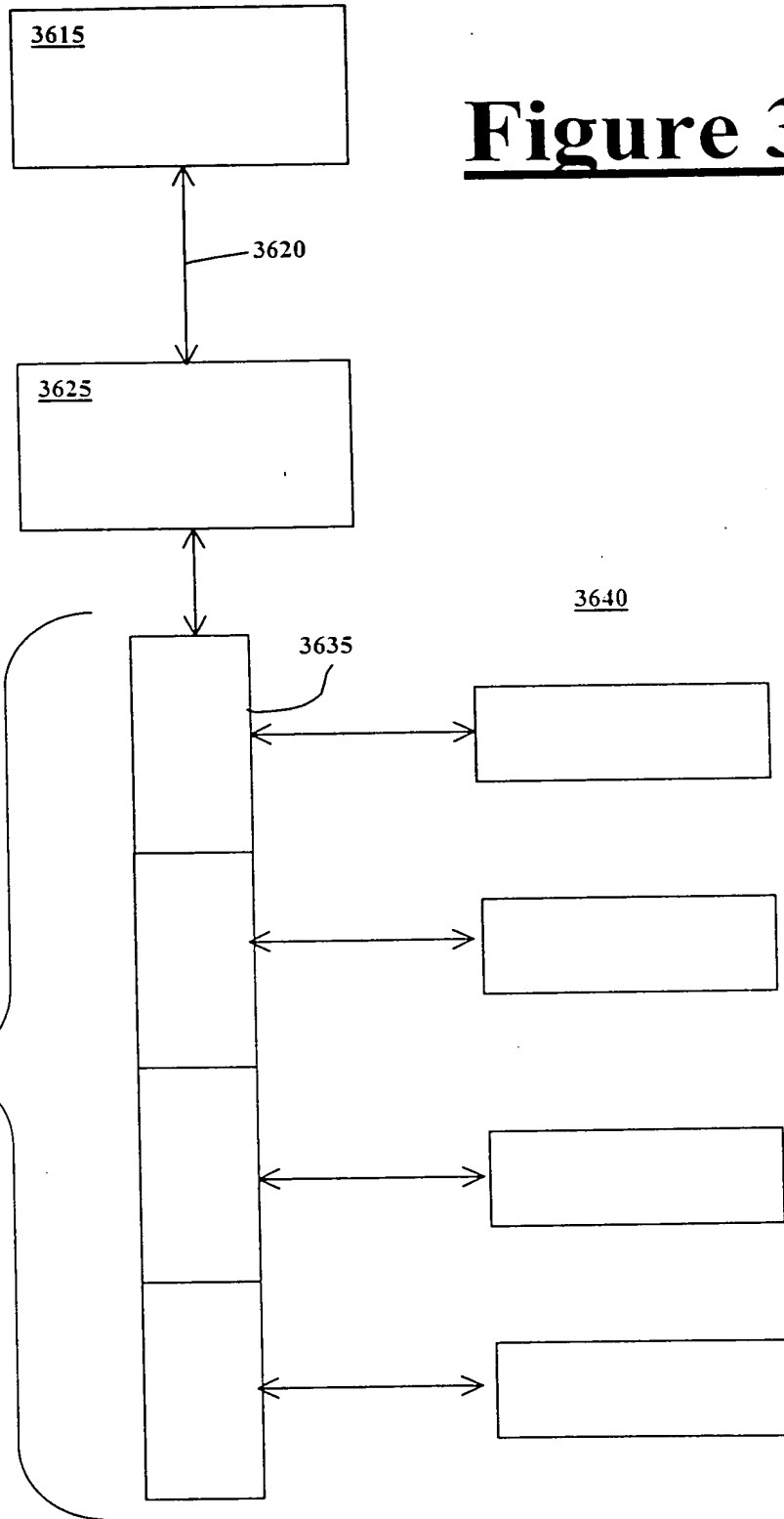
3625

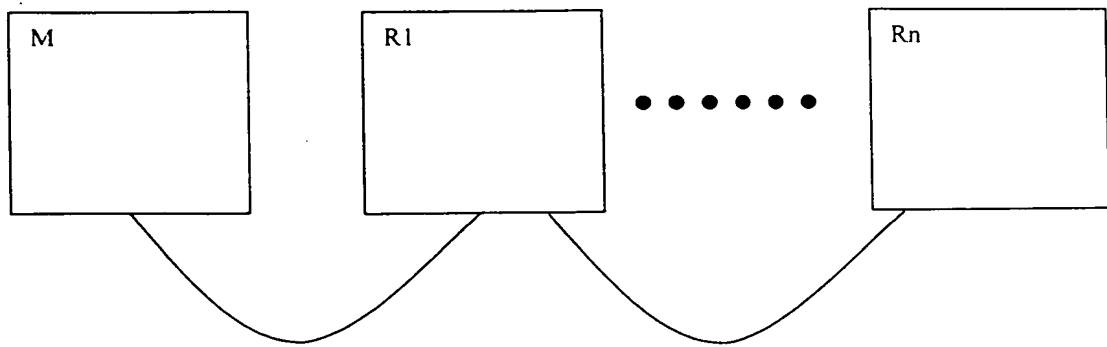
3640

3635

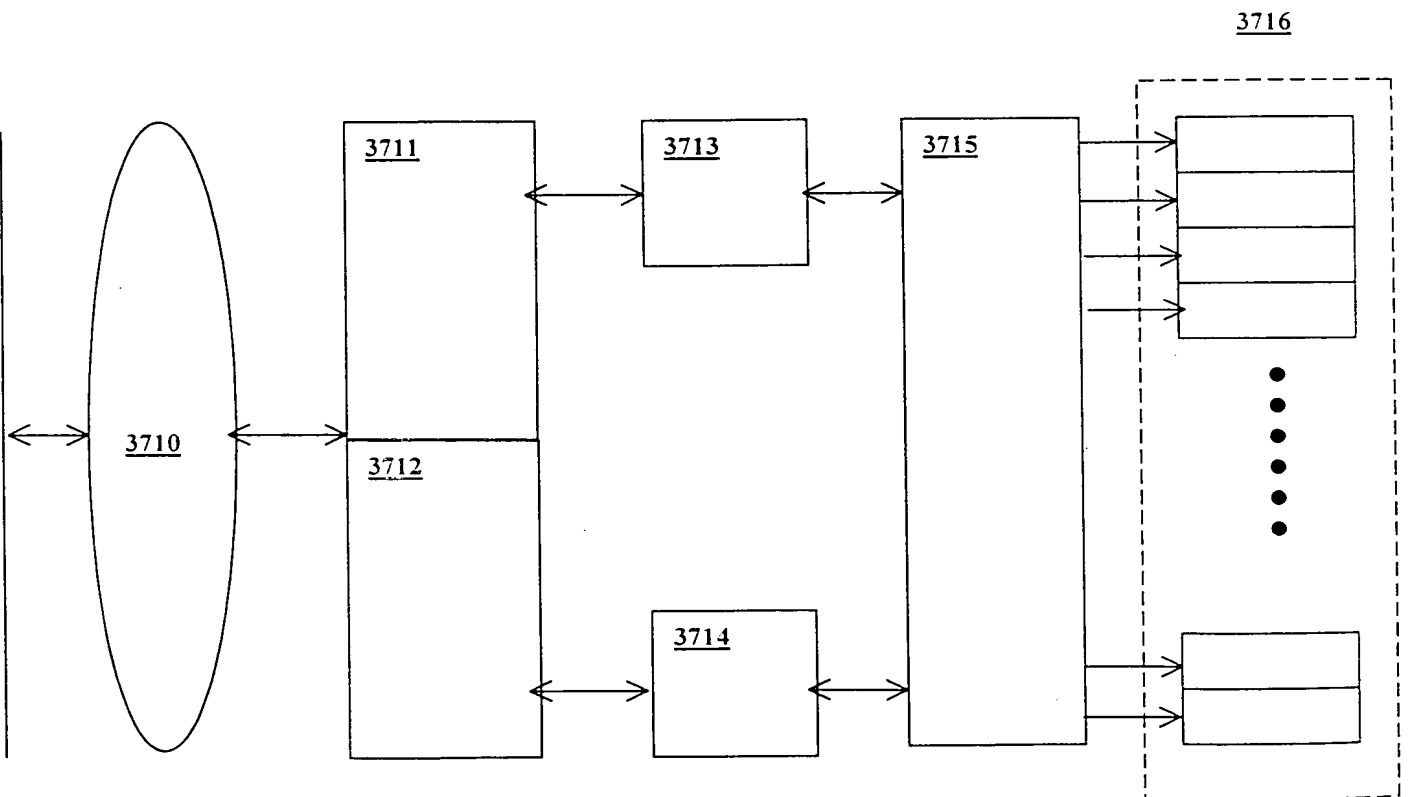
3630

**Figure 36**

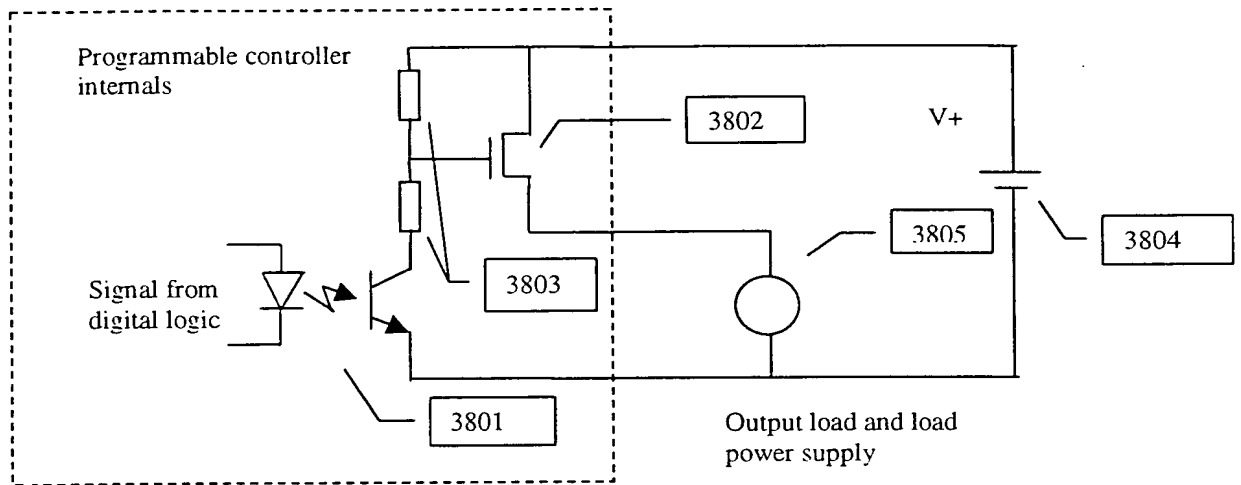




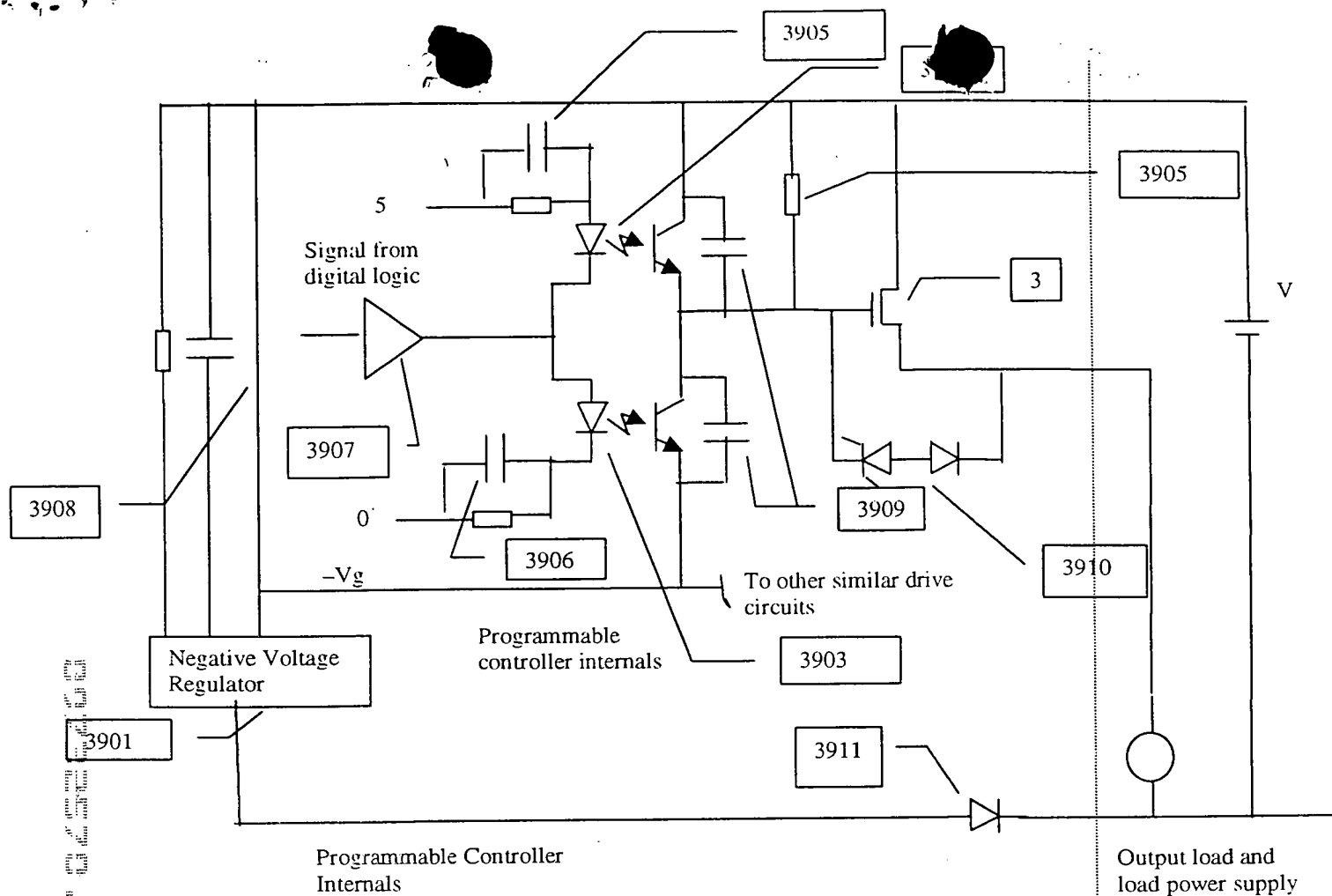
**Figure 37a**



**Figure 37b**



**Figure 38 : Typical DC Output of Programmable Logic Controller  
FET Type Sourcing Output**



**Figure 39 : High speed DC output of Programmable Logic Controller, using push-pull optocoupler circuit.**